L6: Entry 8 of 8 File: USPT Mar 9, 1982

DOCUMENT-IDENTIFIER: US 4319338 A
TITLE: Industrial communications network with mastership determined by need

Detailed Description Text (3):
Various types of controllers may be coupled to the cable 1 through an associated controller interface. These may include general purpose computers 7 coupled to the cable 1 by a computer controller interface 8 or these may include programmable controllers 9 coupled to the cable 1 by a programmable controller interface 10. Similarly, process controllers 11 and numerical control systems 12 may be coupled to the cable 1 by process control interface modules 13 and numerical controller interface modules 14. The types and numbers of controllers will vary with the particular installation, and it is one of the objectives of the present invention to insure flexibility in this respect while maintaining the reliability of the network.

Detailed Description Text (46):
Referring particularly to FIG. 10, the communications rung data stored in the controller interface memory 10 is referred to herein as the command rung index. For each start-done word stored in the controller data table the command rung index stores its 16-bit memory address as indicated at 220. There is no limit to the number of start-done words which can be accommodated, although one or two is usually sufficient. Up to eight command rungs can be associated with each start-done word, and for each of these, a word 222 stores the number of lines the command rung occupies in the controller memory 212 and a word 223 stores a 3-bit pointer and up to five status bits. The rung size words 222 are employed by the scan task 123 to quickly index into the command rungs stored in the controller memory 212 to obtain needed data and the three-bit pointer indicates which one of the eight start bits in the start-done word is associated with that particular command rung. The status bits in the words 223 include a bit which indicates to the scan task 123 that the command rung is the last one associated with a particular start-done word, a bit which indicates that the command rung is the last one in the communications rungs 213, and a bit which indicates to the scan task 123 that the start bit for the command rung has previously been recognized and is being processed.

Detailed Description Text (49):
On the other hand, if a start bit is set, a message is to be sent and the command instruction is read out of the controller memory 212 and examined to determine whether a read, write, or bit control message is to be sent. If data is to be read from a designated station as determined by decision block 228, the message is formed by reading the remaining data in the command rung out of the controller memory 212, attaching a header according to the communication network protocol, and storing it in a message buffer portion of the interface memory 35. Similarly, if a write command is detected at decision block 229 or a bit control command is detected at decision block 230, an appropriate message is formed and stored in a message buffer. As indicated by process block 231, a ROUTE subroutine is then called which is executed to transfer the message to an "output queue" in the memory 35. A listing of the scan and message initiate task 123 is provided in Appendix C.

Detailed Description Text (71):
If the command message is a write command or a bit control command as indicated at decision block 238, the data in the command message is written into the specified location in the controller memory 212. Regardless of the command, a reply message is then formed as indicated at process block 239 and released to the output queue for transmission back to the originating station. If an illegal command is detected at the decision block, an error code is entered in the reply message before it is sent. The format of the reply messages are as follows and a listing of the NETX task is attached as Appendix D.

Detailed Description Text (100):
If a message for this station is received, checks are made to determine if the message was received without transmission errors and that there is memory space available to store the data in the message. If the message can be properly received as determined at decision block 280, the message is "handed over" to the command queue or reply queue as indicated at process block 281. Also, the proper event flag in the process control block is set and the software interrupt is initiated so that the scheduler 125 is entered. If the message was not received properly as determined at decision block 280, a non zero error code is generated and a status message containing the error code is sent back to the originating station. In any case, after sending the status message the system returns to wait for another message.

Detailed Description Text (104):
It should be apparent that there is very little difference between the programmable controller interface 10 and the interface circuits employed to connect other equipment to the communications network. Referring to FIGS. 1, 3 and 6, the difference between the various interface circuits 8, 10, 13 and 14 occurs in the controller driver and receiver circuits 47 and the controller driver routine 117. The serial port provided by the USART 55 on the programmable controller interface 10 is not required on the others, and hence, this hardware and the associated software is eliminated. Almost all commercially available numerical control systems, process control systems and computer control systems provide an RS-232 standard serial port and, therefore, the remaining controller interface circuits 8, 13 and 14 may be virtually identical to one another. With respect to their functioning on the communications network, all of the controller interfaces 8, 10, 13 and 14 operate as described above.

Previous Doc Next Doc Go to Doc#

L6: Entry 1 of 8 File: USPT Mar 13, 2007

DOCUMENT-IDENTIFIER: US 7188928 B2
TITLE: Printer comprising two uneven printhead modules and at least two printer controllers, one of which sends print data to both of the printhead modules

PRIOR-PUBLICATION:

DOC-ID	DATE
US 20060125876 A1	June 15, 2006

Description Paragraph (1198):
If the StatusInResponse register contains "10", this indicates that the application is unable to process the control request. The VCI port's app_stall signal is asserted which causes a STALL handshake to be returned to the USB host.

Description Paragraph (1262):
Possible output functions are 6 Stepper Motor control outputs 18 Brushless DC Motor control output (total of 3 different controllers each with 6 outputs) 4 General purpose LED pulsed outputs. 4 LSS interface control and data 24 Multiple Media Interface general control outputs 3 USB over current protect 2 UART Control and data

Description Paragraph (2327):
When a write command is issued then write_data_accept[1:0] is asserted. This tells the Write Multiplexor that the current write data has been accepted by the DRAM and the write multiplexor can receive write data from the next arbitration winner if it is a write. write_data_accept[1:0] differentiates between CPU and non-CPU writes. A write command is known to have been issued when re_arbitrate_wadv to decide on the next command is detected.

Description Paragraph (2879):
This is the state that NEU enters when a hard or soft reset occurs or when Go has been de-asserted. This state can not left until the reset has been removed, Go has been asserted and it detects that the command controller has entered it's AWAIT_BUFF state. When this occurs the NEU enters the NEU_FILL_BUFF state. ii NEU_FILL_BUFF Before any compressed data can be decoded the NEU needs to fill up its buffer with new data from the SFU. The rest of the LBD waits while the NEU retrieves the first four frames from the previous line. Once completed it enters the NEU_HOLD state. iii NEU_HOLD The NEU waits in this state for one clock cycle while data requested from the SFU on the last access returns. iv NEU_RUNNING NEU_RUNNING controls the requesting of data from the SFU for the remainder of the line by pulsing lbd_sfu_pladvword when the LBD needs a new frame from the SFU. When the NEU has received all the word it needs for the current line, as denoted by the LineLength, the NEU enters the NEU_EMPTY state. v NEU_EMPTY NEU waits in this state while the rest of the LBD finishes outputting the completed line to the SFU. The NEU leaves this state when Go gets deasserted. This occurs when the end_of_line signal is detected from the LBD. 26.3.9 Line Fill Unit Sub-Block Description

Description Paragraph (3459):
The resultant contone pixel is then halftoned. The dither value to be used in the halftoning process is provided by the control data unit. The halftoning process

involves a comparison between a pixel value and its corresponding dither value. If the 8-bit contone value is greater than or equal to the 8-bit dither matrix value a 1 is output. If not, then a 0 is output. This means each entry in the dither matrix is in the range 1-255 (0 is not used).

Description Paragraph (3921):
The PHI can optionally send the calculated fire period by placing META character symbols in a command stream (either a CPU command, or a command configured in the command table). The META symbols are detected by the PHI and replaced with the calculated fire period. Currently 2 META characters are defined.

Description Paragraph (3925):
Immediately after the PHI leaves its reset it will start sending IDLE commands to all printhead data channels. The PHI will not accept any data from the LLU until the Go bit is set. Note the command table can be programmed at any time but cannot be used by the internal PHY when Go is 0.

Description Paragraph (3959):
34.10.7 Command Table

Description Paragraph (3960):
The command table logic contains programmed values for the control symbol lookup table. The print stream controller reads locations in the command table to determine the values of symbols used to construct control commands. The lookup pointers per command are configured by the CmdCfg registers.

Description Paragraph (3961):
The CPU programs the command table by writing to the CmdTable registers. The CPU can write to the command table at any time. But to ensure correct operation of the PHI the CPU should only change the command table when the Go bit is 0.

Description Paragraph (3962):
The command table logic is implemented using a register array (to save logic area). The register array has one read and one write port. The write port is dedicated to the CPU, but the read port needs to be shared between CPU read access and PHI internal read access. To simplify arbitration on the read port, the Go bit is used to switch between CPU access (Go=0) and PHI internal access (Go=1).

Description Paragraph (3975):
On transition into a command state (NCCmd) the command table read address (dc_rd_adr) is loaded with configured start pointer for that command CmdCfg[NC][ST_PTR]. The command could be NC_A or NC_B depending on the value of the segment counter (seg_cnt). While in the command state the dc_rd_adr address is incremented each time a symbol word is written into the output buffer. If the output buffer becomes full the pointer will remain at the current value. While in the NCCmd state the state machine indicates to the symbol mux to select symbols from the command table (ct_ard_data). The state machine determines the command has completed by comparing the dc_rd_adr with the configured end pointer for that command CfgCmd[NC][END_PTR]. If the CfgCmd[NC][EMP] empty bit is set the NCCmd state is bypassed.

Description Paragraph (3978):
When the state machine transitions to the Fire state the command table read address is set to CfgCmd[FIRE][ST_PTR], and the fire_start signal is pulse. The fire_start pulse indicates to the line sync block to update the fire period and dynamic line time minimum value. While in the Fire state the command table address is incremented, and the symbol mux is set to select symbols from the command table (ct_rd_data), and is output to all print channels. The state machine remains in the Fire state until the dc_rd_adr is equal the configured fire command end pointer CmdCfg[FIRE][END_PTR]. When true the state machine transitions back to the Wait state to wait for the next line start pulse. If the CmdCfg[FIRE][EMP] bit is set

the Fire state is bypassed and the state machine transitions from the NozzleData state directly to the Wait state.

Description Paragraph (3984):
The state machine resets to the DataMode state. It allows the data controller state machine control of the symbol mux (sym_sel=dc_sel) and command table (ct_rd_adr=dc_rd_adr).

Description Paragraph (3993):
The symbol mux selects the input symbols and constructs the outgoing data word to the output buffer based on control signals from the mode and data controllers. The input source symbols can come from the the CPU command FIFO, the Data buffer, the Command Table, or from the state machines directly.

Previous Doc Next Doc Go to Doc#

L6: Entry 7 of 8 File: USPT Mar 7, 1995

DOCUMENT-IDENTIFIER: US 5396485 A
TITLE: Information communication system for point-to-point communication and point-to-multipoint communication using same lines

Drawing Description Text (7):
FIGS. 6A and 6B are diagrams showing the structures of the tables 220 and 221 of the configuration control interface shown in FIG. 2A;

Drawing Description Text (8):
FIGS. 7A and 7B are schematic diagrams illustrating the procedure of request and completion processes to be executed by the configuration control interface shown in FIG. 2A;

Drawing Description Text (9):
FIGS. 8A and 8B are schematic diagrams illustrating the procedure of request and completion processes for PTP communication to be executed by the communication control interface shown in FIG. 2A;

Drawing Description Text (11):
FIGS. 10A and 10B are schematic diagrams illustrating the procedure of a data communication process by the control module shown in the figure of the information communication system;

Detailed Description Text (6):
First, the interface between the CPU block 101 and communication interface block 103 of the host computer 100 will be described. The interface between the CPU block 101 and interface block 103 includes a communication control interface 296 and a configuration control interface 298. The communication control interface 296 is used for data communication such as point-to-point (PTP) Communication of data to one terminal 111 and point-to-multipoint (PTMP) communication of same data to a plurality of terminals 111-i (i=1 to n) at a time. The configuration control interface 298 is used for switching between an in-operation (master) system and a stand-by (slave) system of the interface block and for the configuration change of the communication control unit group. Queues or tables to be used by each interface unit are allocated on each main storage unit 102-a, 102-b of the main storage group 102. The communication control interface 296 has the following queues or tables: a request registered destination queue 200 for registering the destination of registration of a request for CPU 101 to be issued to an interface unit 103; PTP communication request queues 201 each for registering a PTP communication request for the data transmission via a single line 109; a PTMP communication request table 202 for registering a PTMP communication request for the transmission of the same data via a plurality of lines; PTP communication completion report queues 203 each for registering the information representative of the state of an execution completion for a PTP communication request by a control module 107; a PTMP communication completion report queue 204 for registering the information representative of the state of an execution completion for a PTMP communication request by an interface unit 103; reception report queues 205 each for registering the information representative of the received data from a terminal 111; communication interrupt report queues 206 each for registering the information representative of an occurrence of abnormality on a line 109; a report registered

destination queue 207 for registering the information representative of a queue to be referred to by a CPU 101 when an interface unit 103 issues an interrupt to a CPU 101 upon the execution completion of a request or upon data reception; PTMP communication management tables 208 each being used by an interface unit 103 for managing the status of data transmission from the interface unit 103 to respective lines during the execution of the PTMP communication request; and PTMP communication request acceptance completion report queue 209 each for registering the information indicating the reception of a PTMP communication request in the queue 241 by a control module 107.

Detailed Description Text (7):
The queues 201, 203, 205, 206, and 209 are provided as many as the number of all lines covered by the communication control unit 105 connected to corresponding interface units 103, so that the order of transmission/reception at each line can be controlled. A plurality of management tables 208 are provided in order for each interface unit 103 to manage a plurality of PTMP communication procedures at a time. Each queue or table of the communication control interface 296 is shared by a pair of interface units 103 of the master and slave, and only the interface unit in the in-operation system is allowed to access it.

Detailed Description Text (8):
In cooperation with the communication control interface 296, the following two work areas are provided in each interface unit 103, namely, a destination pattern table work area 230 and a communication management table edit area 231, The work area 230 is used for storing a destination pattern table for registering the destinations of data for a PTMP communication request issued from a CPU 101 to an interface unit 103. The table edit area 231 is used for editing the PTMP communication management table 208 by using the destination pattern table stored in the work area 230. The edit area 231 is partitioned into as many sub-areas as the number of management tables 208. The management table in the edit area 231 is mainly used for the PTMP communication request process by the interface unit 103, such as searching transmission destinations and generating a transmission completion report to a CPU 101, whereas the management table 208 of the communication control interface 296 is mainly used for recording the history of the PTMP communication request process.

Detailed Description Text (9):
The configuration control interface 298 includes request message tables 220 and completion message tables 221. The request message table 220 is used for reporting a request message such as a system switching message issued from a CPU 101 to an interface unit 103. The completion message table 221 is used for reporting a completion message such as a completion report, indicating the execution completion of a request message, from an interface unit 103 to a CPU 101. The request message/completion message is assigned a plurality of priority levels. A plurality of tables 220 and 221 are therefore prepared for each priority level so as to permit the priority process for a particular request message. Each table of the configuration control interface 298 is independent from the master/slave interface units 103.

Detailed Description Text (26):
Next, referring to FIGS. 7A and 7B, and 8A and 8B, the procedure of the request/completion process by the configuration control interface 298 and communication control interface 296 will be described. First, the request process by the configuration control interface will be described with reference to FIG. 7A. For the control of the interface unit 103 such as system switching, CPU 101: (1) sets request information such as a configuration control command to the request message table 220 of the main storage unit 102, and (2) issues an interrupt to the interface unit 103 via the computer bus 104 to inform the interface unit 103 of the request information. The interface unit 103 received the interrupt from CPU 101: (3) fetches the request information from the message table 220, and (4) issues an interrupt to CPU 101 to inform it of a request information reception completion.

Detailed Description Text (27):
FIG. 7B illustrates the procedure of the completion process by the configuration control interface. The interface unit 103 which completed the execution of the request information received during the request process, reports the completion of the process to CPU 101 as in the following. The interface unit 103: (1) sets completion report information to the completion message table 221 of the main storage unit 102, and (2) issues an interrupt to CPU 101 to report the completion of the process to CPU 101. CPU 101 received the interrupt from the interface unit 103: (3) fetches the completion information of the completion message table 221, and (4) issues an interrupt to the interface unit 103 to inform the interface unit 103 of the completion of the reception of the completion report.

Detailed Description Text (28):
FIGS. 8A and 8B show the outline of the PTP communication request/completion process contained in the communication process to be executed by the communication control interface and the interface between the communication control unit and control module. FIG. 8A shows the procedure of the PTP communication request process by the two interfaces. In requesting for data transmission to a desired single line 109 via the control module 107, CPU 101: (1) registers a PTP communication request in the PTP communication request queue 201 of the main storage unit 102 corresponding to the desired line 109. Next, CPU 101: (2) issues the communication request by registering the information for identifying the queue 201 in the request registered destination queue 200 of the main storage unit 102. The interface unit 103: (3) periodically monitors the information registered by CPU 101 in the request registered destination queue 200, and when the registered information is detected, (4) registers a PTP communication command in the queue 240 of the control module 107. The PTP communication command contains the communication type representing whether communication is a PTP communication or a PTMP communication, and the storage address and size of transmission data. The PTMP communication command contains, in addition to the communication type and the storage address and size of transmission data, the serial number of the destination pattern table 500 used by the PTMP communication.

Detailed Description Text (34):
In this manner, the in-operation interface unit performs the request process in response to the data communication request from CPU via the communication control interface, only when the interface unit is in the master state.

Detailed Description Text (35):
FIG. 10A shows the procedure of the PTP communication request process by the control module 107. The control module 107 periodically monitors the command registered by the interface unit 103. As described with FIG. 9, if the control module 107: (1) detects the PTP communication command registered in the PTP communication command queue 240, the control module 107: (2) transfers, in accordance with the contents of the detected command, the transmission data from the transmission/reception buffer 114 to the buffer 116, and thereafter (3) transmits the data to the single line 109.

Detailed Description Text (38):
FIG. 10B shows the procedure of the process of receiving data from a terminal 111 to be executed by the control module 107. In reporting a data reception from a terminal to the interface unit 103, the control module 107: (1) transfers received data from the buffer 116 to the buffer 114, (2) registers a reception report in the reception report queue 205 of the communication control interface 296, and (3) sets information to the notice table 242 to issue an interrupt to the interface unit 103.

Detailed Description Text (43):
FIG. 12 is a flow chart explaining the PTMP communication request process by an in-

operation interface unit. The interface unit 103 switches the request process 900 to the PTMP communication request process 911 when the communication type is judged as the PTMP communication from the registration destination identification information supplied from CPU 101. In this process 911, in accordance with the identification information, the interface unit 103 reads the PTMP communication request from the area 322 of the PTMP communication request table 202, and transfers the transmission data from the transmission/reception buffer 114 to the buffer 115 in accordance with the contents of the PTMP communication request (Step 1201). The destination pattern table 500 designated by CPU 101 when the PTMP communication request was issued is copied from the work area 230 to the request management area 512 and completion monitor area 514 corresponding to the area 322 of the edit area 231, to form a management table for the PTMP communication (Step 1202). The table is then registered in the management queue 208 of the communication control interface 296 (Step 1203). Next, a destination line for the transmission data is detected from the request management area 512 of the edit area 231 (Step 1204), and a PTMP communication command is written at the area 333 of the command queue 241 indicated by the write pointer (Step 1205). The interface unit 103 checks whether the command was correctly written (Step 1206). If written correctly, the write pointer 331 in the command queue 241 is updated (Step 1207), and the line request flag in the request management area 512 of the management table 208 is changed from "1" to "0" (Step 1208). The write pointer 331 of the command queue 241 is stored in the line record element 402 of the control module field 404 which is contained in the PTMP communication record area 402 in the current write pointer table 210 of the main storage unit 102 (Step 1209). As a result of the check of the step 1206, if not written correctly, the serial number of the control module :not registered, the line number, and the abnormality type, are stored in the serial number record field 516 for the control module, the line number record field 517, and the abnormality type record field 518, respectively of the abnormality completion record area of the management table (Step 1210). The abnormality completion count 515 of the management table is incremented by "1" (Step 1211), and the line monitor flag of the completion monitor area 514 is changed from "1" to "0". The completion report still-not-received count 513 is decremented by "1", and the line request flag of the request management area 512 is changed from "1" to "0" (Step 1212). After the above processes, the request still-not-accepted count 511 of the management table is decremented by "1" (Step 1213) and it is judged whether the request still-not-accepted count 511 is "0" (Step 1214). If the count 511 is not "0", the next destination line for the transmission data is detected from the request management area 512 (Step 1204), and the above operations are repeated. If the judgement result at Step 1214 indicates that the count 511 is "0", it means that there is no line 109 to which the PTMP transmission data is sent. In order to monitor the acceptance completion report for the PTMP communication request from the control module 107, the interface unit 103 activates a completion monitor timer (Step 1215), updates the read pointer 302 of the request registered destination queue 200, and then discards the destination identification information to terminate the procedure. In this manner, the in-operation interface unit 103 fetches the PTMP transmission data from the main storage unit, and registers the PTMP communication command in the command queue 241 of each control module connected with data transmission lines, in accordance with the contents of the request management area 512 of the management table 208 generated from the destination pattern table 500.

Detailed Description Text (59):
As described previously, the master/slave state of the interface unit 103 is determined from the state of the interface channels 117. The transition of the state of the interface unit 103 can be executed by a command of changing the state of the interface unit 103, namely, by a channel connection/disconnection command for instructing the connection/disconnection between the internal bus controller 106 connected to the channel 117 of the interface unit 103 and the internal bus 108. This command is transferred via the table 220, 221 of the configuration control interface 298.

Detailed Description Text (60):
FIG. 16 shows the request information set in the request information setting area of the request message table 220 of the configuration control interface 298 of the main storage unit 102 when the channel connection/disconnection command is requested. In FIG. 16, reference numeral 161 represents a command ID indicating whether the command is for the channel connection/disconnection, reference numeral 162 represents a sub command ID indicating whether the contents of the command ID 161 indicate the connection/disconnection of all the channels 117 of the interface unit 103 or of a particular channel 117, reference numeral 163 represents a channel address of the particular channel 117 requested by the sub command ID 162, this channel address being effective only when the sub-command ID 162 requests for the connection/disconnection of the particular channel 117, and reference numeral 164 represents a data integrity request flag indicating a presence/absence of the communication process inheritance request at the original master state when the interface unit 103 is changed from the slave state to the master state by a channel connection command. In the switching of the interface unit 103 between the in-operation system and the stand-by system, the sub command ID 162 requests for the connection/disconnection of either all the channels or of the particular channel such as when one communication control unit 105 is added or replaced. The request flag 164 is therefore effective only when the sub command ID 162 requests for all the channels at the channel connection command.

Detailed Description Text (62):
FIG. 17 is a flow chart explaining the operation when an interface unit receives the channel connection command from CPU 101 via the configuration control interface 298. Upon reception a request from CPU 101, the interface unit 103 fetches the contents of the request information setting area 602 of the message table 220. The interface unit 103 recognizes a request of the channel connection command from the command ID 161, and executes a channel connection process 1700. In this process 1700, the sub command ID 162 is first checked (Step 1701). If the sub command ID 162 indicates a request for all the channels, the internal bus controllers 106 presently connected to all the channels are logically disconnected from the internal busses (Step 1702), and a bus transfer inhibit interrupt is issued to all the control modules 107 of the communication control units 105 connected to the channels 117 (Step 1703). Then, the internal bus controllers 106 connected to all the channels 117 are logically connected to the internal busses 108 (Step 1704), to change the state of the interface unit 117 to the master state (Step 1705). Thereafter, the request flag 164 is checked (Step 1706). If the data integrity is requested, an in-operation system inheritance process is executed (Step 1707) to release the bus transfer inhibit of all the control modules 107 of the communication control units 105 connected to the channels 117. In this case, no operation is executed if the interface unit 103 received the channel connection command for all the channels is in the master state. On the other hand, if the sub-command ID 162 indicates a request for the connection to a particular channel, the internal bus controller 106 connected to the particular channel 117 indicated by the channel address 163 is logically connected to the internal bus 108 (Step 1709). The state of the interface unit 103 is then checked (Step 710). If in the slave state, it is changed to the master state (Step 1711).

Detailed Description Text (63):
FIG. 18 is a flow chart explaining the operation when an interface unit receives the channel disconnection command from CPU 101 via the configuration control interface 298. Upon reception of a request from CPU 101, the interface unit 103 fetches the contents of the request information setting area 602 of the message table 220. The interface unit 103 recognizes a request of the channel disconnection command from the command ID 161, and executes a channel disconnection process 1800. In this process 1800, the sub command ID 162 is first checked (Step 1801). If the sub command ID 162 indicates a request for the disconnection of all the channels, the internal bus controllers 106 presently connected to all the channels are

logically disconnected from the internal busses (Step 1802), to change the state of the interface unit 103 to the slave state (Step 1803). If the sub command ID 162 indicates a request for the disconnection of a particular channel, the internal bus controller 106 connected to the channel 117 indicated by the channel address 163 is logically disconnected from the internal bus 108 (Step 1804). The states of other channels are checked (Step 1805). If there is a reserved channel, the state of the interface unit 103 is changed to the slave state (Step 1806). In this case, no operation is executed if the interface unit 103 received the channel disconnection is in the slave state. The interface unit 103 received the channel disconnection command and changed from the master state to the slave state intercepts all the internal communication processes, and inhibits thereafter receiving the request from the communication control interface 298 and an interrupt, as described previously with FIGS. 9 and 11.

Detailed Description Text (69):
FIG. 21 is a flow chart showing the entirety of the in-operation system process inheritance process. In the in-operation system process inheritance process 1707, a process is first executed wherein the communication control information used by the PTMP communication process described previously is recovered to the interface unit 103 (Step 2101). Next, a recovery process is executed for inheriting the process of receiving an interrupt from the control module 107 of the communication control unit 105 (Step 2102). Then, a recovery process is executed for inheriting the process of accepting a request from CPU 101 via the communication control interface 296 (Step 2103). Lastly, a recovery process is executed for inheriting the completion report process of the PTMP communication process (Step 2104).

Detailed Description Text (70):
FIG. 22 is a flow chart showing a process 2101 of recovering the communication control information used by the PTMP communication process to the communication control interface 296. In this recovery process 2101, all the destination pattern tables 500 in the storage area 212 of the recovery interface 294 in the main storage unit 102 are copied to the work area 230 of the interface unit 103 (Step 2201). Next, It is checked, from the request flags 321 of the PTMP communication request table 202 of the communication control interface 296, whether there is a valid PTMP communication request (Step 2202). If there is an effective PTMP communication request, it is checked whether the PTMP communication process is in the status 2 (Step 2203). If in the status 2, the management table 208 of the interface 296 is copied to the edit area 231 of the interface unit 103 (Step 2204). If not, it is not copied. If there is no effective communication request, no operation is performed. The transmission data for the PTMP communication request is copied from the transmission/reception buffer 114 to the buffer 115 (Step 2205). Of the above process, Steps 2202 to 2205 are repeated as many times as the number of information registration areas of the request table 202 (Step 2206). In this manner, only the information at the in-operation system necessary for the PTMP communication process can be recovered to the stand-by system interface unit prior to executing the actual process inheritance.

X

Detailed Description Text (72):
FIG. 24 is a flow chart showing a recovery process of inheriting the process of accepting a request from CPU 101 via the communication control interface. In this recovery process 2103, it is checked, from the read pointer 302 of the destination queue 200 of the communication control interface 296 and the value of the queue 2000 record field 410 of the next read pointer table 211 of the recovery interface 294, whether the in-operation interface unit 103 has been operating in the command acceptance process via the communication control interface 296 (Step 2401). If in the command acceptance process (if the read pointer 302 is not coincident with the next read pointer), the destination identification information is checked which is registered in the information registration area 303 of the read pointer 302 of the queue 200 (Step 2402). The following request inheritance process is performed thereafter. If the destination identification information indicates the PTP

communication request, it is checked, from the write pointer 311 of the PTP communication command queue 240 of the control module 107 identified by the identification information and the current write pointer of the current write pointer table 210 of the recovery interface 294, whether the PTP communication command in the area 313 indicated by the read pointer 312 of the request queue 201 identified by the identification information has been registered in the command table 240 (Step 2403). If not (if the write pointer 311 is coincident with the current write pointer), the PTP communication command is registered in the command queue 240 (Step 2406), and the next read pointer in the request queue 201 of the next read pointer table 211 is updated (Step 2407). If the PTP communication command has been registered in the command queue 240, it is checked, from the read pointer 312 of the request queue 201 and the next read pointer of the next read pointer table 211, whether the next read pointer of the request queue 201 has been updated (Step 2404). If not, Step 2407 is executed and thereafter, the identification information in the queue 200 is discarded (Step 2405). If the identification information indicates the PTMP communication, it is checked whether the PTMP communication process is in the PTMP status 1 (Step 2408). If in the PTMP status 1, after the completion of the in-operation system process inheritance process 1707, the PTMP communication request process 1200 is requested to be executed (Step 2409). If not in the PTMP status 1, the recovery process for the PTMP communication request is executed (Step 2410). In this manner, various types of request from CPU 101 via the communication control interface can be executed without any fail or duplication. The request acceptance process at the communication control interface is limited only to one request so that it is sufficient for the request acceptance recovery process to deal with the one request at a time.

Detailed Description Text (73):
FIG. 25 is a flow chart showing a recovery process 2410 of inheriting the PTMP communication request process to the control module of a communication control unit during the process of accepting a PTMP communication request from CPU 101 via the communication control interface 296. In this recovery process 2410, a line expected to be in the PTMP communication request process is detected in accordance with the request still-not-accepted count 511 in the management table at the edit area 231 of the interface unit 103 and the number of requested lines obtained from the destination pattern table 500 (Step 2501). It is checked whether the request flag for the detected line in the management table at the request management area 512 has been changed (Step 2502). If not (if the request flag is "1"), it is checked, from the from the write pointer 321 of the command queue 241 of the control module 107 covering the line and the current pointer in the PTMP communication record area 402 at the line record element 406 of the current write pointer table 210 of the recovery interface 294, whether the PTMP communication command has been registered in the command queue 241 (Step 2503). If the command has not been registered (if the write pointer 331 is coincident with the current write pointer), the PTMP communication command is registered in the command queue 241 and the request flag is changed to "0" (Step 2504). The write pointer 331 of the command queue 241 is stored in the current write pointer table 210 at the line record element 406 in the PTMP communication record area 402 (Step 2505), and thereafter the request still-not-accepted count 511 of the management table is decremented by "1" (Step 2506). The PTMP communication request process (Steps 1205 to 1214 of the PTMP communication request process 1200) is repeated for the remaining requested lines (Step 2507) until the request still-not-accepted count 511 in the management table becomes "0" (Step 2508). The identification information of the destination is discarded from the queue 200 (Step 2509). After the completion of the in-operation system process inheritance process 1707, a monitor timer for monitoring the completion of the PTMP communication process is started (Step 2510). In this manner, in the PTMP communication process during the request process at each control module of the communication control unit designated by the interface unit, data transmission to lines can be inherited without any fail or duplication.

threshold

Detailed Description Text (74):
FIG. 26 is a flow chart showing a recovery process 2104 of inheriting the completion report process in the PTMP communication process. In this recovery process 2104, whether there is an effective communication command is checked from the request flags 321 of the request queue 202 of the communication control interface 296 (Step 2601). If there is an effective communication command, it is checked whether the PTMP communication process is in the PTMP status 2 (Step 2602). If in the PTMP status 2, it is checked whether the non-completion report count 513 of the management table at the edit area 231 of the interface unit 103 is "0" (Step 2603). If "0", ill is checked whether the completion report information for the management table whose completion report still-not-received count 513 is "0" is contained in the effective completion report information registered in between the area 313 defined by the write pointer 311 and read pointer 312 of the completion report queue 204 (Step 2604). If not, the completion report information for the PTMP communication process is generated from the management table and registered in the completion report queue 204 (Step 2605). The identification information of the interrupt report destination for the PTMP communication completion report is registered in the report registered destination queue 207 (Step 2606), and an interrupt is issued to CPU 101 (Step 2607). Steps 2601 to 2607 are repeated as many times as the number of information registration areas of the request queue 202 (Step 2608). In this manner, the PTMP communication completion report process for CPU 101 can be inherited.

CLAIMS:

17. An information communication system according to claim 16, wherein:

said specific main storage unit includes a recovery area for storing inheritance information,

said old master interface unit includes means for suspending processes for requests from said CPUs and making said control modules suspend reporting processes responsive to the requests,

said new master interface unit includes means for resuming said suspended processes for the requests in accordance with said inheritance information, and restarting the reporting processes of said control modules.

Previous Doc Next Doc Go to Doc#

Searches for User *jcorrielus1* (Count = 19604)

Queries 19555 through 19604.

Find

Latest Prev Next Oldest

Edit Help Cancel

S #	Updt	Database	Query	Time	Comm
S19604	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	5396485.pn.	2007-07-06 19:41:56	
S19603	U	USPT	(process near control and (command near table) and (detect$ near command)) and (control near interface)	2007-07-06 18:07:32	
S19602	U	USPT	(process near control and (command near table) and (detect$ near command)) and and (control near interface)	2007-07-06 18:07:14	
S19601	U	USPT	(process near control and (command near table)) and (detect$ near command)	2007-07-06 18:05:54	
S19600	U	USPT	(process near control) and (command near table)	2007-07-06 18:05:38	
S19599	U	USPT	process near control	2007-07-06 18:04:57	
S19598	U	USPT	((5491531 or 5553095 or	2007-07-06	

			5537549).pn.) and (command near table)	18:02:21
S19597	U	USPT	(5491531 or 5553095 or 5537549).pn.	2007-07-06 17:45:33
S19596	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(707/$.ccls. and (database near management) and (bucket$) and segment$ and correlat$) and (storage near area)	2007-07-06 09:15:25
S19595	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(707/$.ccls. and (database near management) and (bucket$) and segment$) and correlat$	2007-07-06 09:15:02
S19594	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(707/$.ccls. and (database near management) and (bucket$)) and segment$	2007-07-06 09:14:22
S19593	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(707/$.ccls. and (database near management)) and (bucket$)	2007-07-06 09:14:05
S19592	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(707/$.ccls.) and (database near management)	2007-07-06 09:13:29
S19591	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	707/$.ccls.	2007-07-06 09:12:45
S19590	U	USPT	Adaptive near hashing	2007-07-05 16:59:16
S19589	U	USPT	(manufact$ near controller and (protocol or command$) and (controller near device) and (exchang$ near data) and (real near time)) and	2007-07-05 16:14:52

			(detect$ near command)	
S19588	U	USPT	(manufact$ near controller and (protocol or command$) and (controller near device) and (exchang$ near data)) and (real near time)	2007-07-05 16:14:11
S19587	U	USPT	(manufact$ near controller and (protocol or command$) and (controller near device) and (exchang$ near data)) and (reralt near time)	2007-07-05 16:13:57
S19586	U	USPT	(manufact$ near controller and (protocol or command$) and (controller near device)) and (exchang$ near data)	2007-07-05 16:13:28
S19585	U	USPT	(manufact$ near controller and (protocol or command$)) and (controller near device)	2007-07-05 16:12:56
S19584	U	USPT	(manufact$ near controller) and (protocol or command$)	2007-07-05 16:12:25
S19583	U	USPT	manufact$ near controller	2007-07-05 16:12:06
S19582	U	USPT	manufacturer near controller near device	2007-07-05 16:11:48
S19581	U	USPT	manutacturer near controller near device	2007-07-05 16:11:33

S19580	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(707/6.ccls. and (DIAGNOSTIC NEAR REPORT)) and sort$	2007-07-05 15:31:21
S19579	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(707/6.ccls.) and (DIAGNOSTIC NEAR REPORT)	2007-07-05 15:28:33
S19578	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(707/6.ccls.) and (assessment near model)	2007-07-05 15:28:07
S19577	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(707/6.ccls.) and (assessment near model)	2007-07-05 15:27:35
S19576	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	707/6.ccls.	2007-07-05 15:27:08
S19575	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(assistant near platform) and (user near quer$)	2007-07-05 14:34:11
S19574	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(assistant near platform) and (diagnostic near report)	2007-07-05 14:33:30
S19573	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	assistant near platform	2007-07-05 14:32:39
S19572	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	unobtrusive near logging	2007-07-05 14:25:18
S19571	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	unobtrusive near workflow	2007-07-05 14:24:26
S19570	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	6850891.pn.	2007-07-03 15:57:26
S19569	U	USPT	7080328.pn.	2007-07-03 13:49:51
S19568	U	USPT	7080328.pn.	2007-07-03 12:34:54
S19567	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	((STAR NEAR	2007-

			HYPERBOLIC NEAR TREE)) AND CONVERT$	07-03 11:54:53
S19566	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(STAR NEAR HYPERBOLIC NEAR TREE)	2007-07-03 11:54:31
S19565	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(patent near analysis) and xml	2007-07-03 11:49:48
S19564	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	patent near analysis	2007-07-03 11:49:11
S19563	U	USPT	(((first near application) same (second near application)) and (map$ near data)) and ((translat$ or convert$) near (first near application))	2007-07-02 12:33:42
S19562	U	USPT	(((first near application) same (second near application))) and (map$ near data)	2007-07-02 12:32:28
S19561	U	USPT	((first near application) same (second near application))	2007-07-02 12:32:13
S19560	U	USPT	((first near application) and (second near application)) and (map$ near ((first near application) same (second near application)))	2007-07-02 12:30:42
S19559	U	USPT	((first near application) and (second near	2007-07-02 12:26:59

			application)) and (map$ near ((first and second) near application))	
S19558	U	USPT	(first near application) and (second near application)	2007-07-02 12:25:58
S19557	U	USPT	((convert$ or translat$) near (data near structure) same (database near table$))	2007-07-02 12:05:24
S19556	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(expert near choice and (data near grid)) and map$	2007-07-02 06:38:02
S19555	U	PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD	(expert near choice) and (data near grid)	2007-07-02 06:37:49

Find

Latest Prev Next Oldest

Edit Help Cancel